Question 1:

String Permutations

Given two Strings 'X' and 'Y'(each containing at most 1000 characters),find if X is a permutation of Y, i.e We can rearrange the characters in X to form Y.

The Output should be "yes", if X is a permutation of Y , "no", if not.

Input Specification:

input1 : the string 'X'

input2 : the string 'Y'

Output Specification:

Return "yes" or "no" accordingly

Example 1:

input1 : ab

input2 : ab

Solution:

package com;

import java.util.Scanner;

public class StringPermutations {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

System.out.println("Enter a String :");

String str=sc.next(); //"ABC";

int n=str.length();

StringPermutations permutations=new StringPermutations();

System.out.println("Permutations of Given String :");

permutations.permute(str, 0, n-1);

}

public void permute (String str,int l,int r)

{

if(l==r)

System.out.println(" "+ str);

else {

for(int i=l;i<=r;i++) {

str=swap(str,l,i);

permute(str, l+1, r);

str=swap(str, l, i);

}

}

}

public String swap(String a,int i,int j) {

char temp;

char[] charArray=a.toCharArray();

//swap

temp=charArray[i];

charArray[i]=charArray[j];

charArray[j]=temp;

return String.valueOf(charArray);

}

}

Question 2:Question :

Highest Common Factor

find the HCF(Highest Common Factor) of n numbers given in an Integer Array.

Fill in the function HCF() and return the HCF.

Input Specification :

Input1: the size array.

Input2: an integer array

output Specification :

Return the HCF of given numbers.

Example 1:

input1: 3

input2:(2,4,8)

Solution:

package com;

import java.util.Scanner;

public class HCF {

static int gcd(int a, int b) {

if(b==0)

return a;

return gcd(b,a%b);

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

int input1,input2;

System.out.println("Enter a input1 & input2 :");

input1=sc.nextInt();

input2=sc.nextInt();

System.out.println("GCD of "+input1+" and "+input2+" is :"+gcd(input1,input2));

}

}

Question 3:Java Regex 2 - Duplicate Words

In this challenge, we use regular expressions (RegEx) to remove instances of words that are repeated more than once, but retain the first occurrence of any case-insensitive repeated word. For example, the words love and to are repeated in the sentence I love Love to To tO code. Can you complete the code in the editor so it will turn I love Love to To tO code into I love to code?

To solve this challenge, complete the following three lines:

Write a RegEx that will match any repeated word.

Complete the second compile argument so that the compiled RegEx is case-insensitive.

Write the two necessary arguments for replaceAll such that each repeated word is replaced with the very first instance the word found in the sentence. It must be the exact first occurrence of the word, as the expected output is case-sensitive.

Note: This challenge uses a custom checker; you will fail the challenge if you modify anything other than the three locations that the comments direct you to complete. To restore the editor's original stub code, create a new buffer by clicking on the branch icon in the top left of the editor.

Input Format

The following input is handled for you the given stub code:

The first line contains an integer, , denoting the number of sentences.

Each of the subsequent lines contains a single sentence consisting of English alphabetic letters and whitespace characters.

Constraints

Each sentence consists of at most English alphabetic letters and whitespaces.

Output Format

Stub code in the editor prints the sentence modified by the replaceAll line to stdout. The modified string must be a modified version of the initial sentence where all repeat occurrences of each word are removed.

Sample Input

5

Goodbye bye bye world world world

Sam went went to to to his business

Reya is is the the best player in eye eye game

in inthe

Hello hello Ab aB

Sample Output

Goodbye bye world

Sam went to his business

Reya is the best player in eye game

in inthe

Hello Ab

Explanation

* We remove the second occurrence of bye and the second and third occurrences of world from Goodbye bye bye world world world to get Goodbye bye world.
* We remove the second occurrence of went and the second and third occurrences of to from Sam went went to to to his business to get Sam went to his business.
* We remove the second occurrence of is, the second occurrence of the, and the second occurrence of eye from Reya is is the the best player in eye eye game to get Reya is the best player in eye game.
* The sentence in inthe has no repeated words, so we do not modify it.
* We remove the second occurrence of ab from Hello hello Ab aB to get Hello Ab. It's important to note that our matching is case-insensitive, and we specifically retained the first occurrence of the matched word in our final string.

Solution:

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class DuplicateWords

{

public static void main(String[] args){

String pattern = "(\\b\\w+\\b)(\\s\*\\1\\b)+";

Pattern r = Pattern.compile(pattern, Pattern.CASE\_INSENSITIVE);

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

while(testCases>0){

String input = in.nextLine();

Matcher m = r.matcher(input);

boolean findMatch = true;

while(m.find( )){

input = input.replaceAll(m.group(), m.group(1));

findMatch = false;

}

System.out.println(input);

testCases--;

}

}

}

Question 4:

Java program to display prime numbers from 1 to 100 and 1 to n?The number which is only divisible by itself and 1 is known as prime number.For example 2, 3, 5, 7…are prime numbers.

Solution:

class PrimeNumbers

{

public static void main (String[] args)

{

int i =0;

int num =0;

//Empty String

String primeNumbers = "";

for (i = 1; i <= 100; i++)

{

int counter=0;

for(num =i; num>=1; num--)

{

if(i%num==0)

{

counter = counter + 1;

}

}

if (counter ==2)

{

//Appended the Prime number to the String

primeNumbers = primeNumbers + i + " ";

}

}

System.out.println("Prime numbers from 1 to 100 are :");

System.out.println(primeNumbers);

}

}